A Copy Mechanism for Handling Knowledge Base Elements in SPARQL Neural Machine Translation

Rose Hirigoyen, Amal Zouaq and **Samuel Reyd**
rose.hirigoyen@polymtl.ca
amal.zouaq@polymtl.ca
samuel.reyd@polymtl.ca
Polytechnique Montreal

Abstract

Neural Machine Translation (NMT) models from English to SPARQL are a promising development for SPARQL query generation. However, current architectures are unable to integrate the knowledge base (KB) schema and handle questions on knowledge resources, classes, and properties unseen during training, rendering them unusable outside the scope of topics covered in the training set. Inspired by the performance gains in natural language processing tasks, we propose to integrate a copy mechanism for neural SPARQL query generation as a way to tackle this issue. We illustrate our proposal by adding a copy layer and a dynamic knowledge base vocabulary to two Seq2Seq architectures (CNNs and Transformers). This layer makes the models copy KB elements directly from the questions, instead of generating them. We evaluate our approach on state-of-the-art datasets, including datasets referencing unknown KB elements and measure the accuracy of the copy-augmented architectures. Our results show a considerable increase in performance on all datasets compared to non-copy architectures.

1 Introduction

The Semantic Web organizes concepts in optimized, machine-readable, knowledge bases (KB) (or knowledge graphs). Still, as these knowledge bases are not immediately designed with a human user in mind, the SPARQL Protocol and RDF Query Language (SPARQL) is hardly accessible to laypeople with little-to-no knowledge of programming languages. This creates a strong accessibility bias, as it prevents users from accessing sizeable amounts of information because of their lack of a specific skillset.

One way to bypass any need for prior knowledge is by allowing the users to query KBs using natural language questions. Figure 1 illustrates the task at hand. More specifically, using neural machine translation (NMT) to translate natural language questions to SPARQL queries has proven to be an interesting avenue to solve this challenge, with BLEU-score performances of more than 90% across multiple datasets (Yin et al., 2021).

Q: What is Villa La Mauresque ?

```
select ?a where
    { dbr:Villa_La_Mauresque
    dbo:abstract ?a }
```

A: The villa La Mauresque is located in cap Ferrat (Alpes-Maritimes) and was remodeled in 1927 ...

Figure 1: Example of the SPARQL NMT task

However, behind these high-performing architectures are models that rarely return the correct answer to a question about a topic they have never seen in training, even if the information is available in the KB. As a single wrong answer can negatively affect the user's trust in the model, this limitation becomes a critical downfall for an automatic SPARQL query generation model. The main goal of this paper is to propose a mechanism to effectively generate accurate SPARQL queries. In particular, we aim at handling out-of-vocabulary (OOV) knowledge base elements at the schema level (classes, properties) and the instance level. As such, we put forth the following research questions:

- **RQ1:** Is the integration of the KB elements in the question sufficient for the model to handle OOV elements?
- **RQ2:** Is the accuracy of the translations improved if the neural translation architecture is able to copy KB elements directly from the question?
- **RQ3:** Does the evaluation of the model on a dataset composed solely of unknown KB

Findings of the Association for Computational Linguistics: AACL-IJCNLP 2022, pages 226–236
November 20–23, 2022.

elements allows for a complete overview of the model's capabilities?

Our main contributions are as follows. (1) Given a working tagging algorithm, we propose a way to allow NMT models to handle questions on topics they have not seen during training. (2) We propose a methodology to evaluate a model's performance exhaustively. (3) Finally, we produce standardized, corrected, and tagged versions of the datasets to foster reproducibility and future developments in this research field[1].

2 Related Work

Knowledge Bases Terminology. A *knowledge base (KB)* stores data in the form of one or more Resource Description Framework (RDF) graphs, in which the nodes are concepts or instances, and the edges encode the relationship between them. An RDF graph is described using (subject, property, object) triples, which we refer to as *KB elements*. Each KB element has a unique URI, which is used to reference it in a SPARQL query and a label, which is their name in a natural language. If there is no label, we can generate one from the element's URI.

Seq2Seq for NMT. The base architecture behind many NMT models is *Seq2Seq*, which learns to generate words using source and target vocabularies. If there is a token in the source sentence that is not in the vocabulary, the model simply replaces it by the **<unk>** placeholder token. The model is as such only able to generate tokens that are in its target vocabulary. The transformers (Vaswani et al., 2017) and convolutional networks (Gehring et al., 2017) are currently the two best non-pretrained architectures for SPARQL NMT, as reported by Yin et al. (2021).

As more data becomes available, an important development in this field is the introduction of pretrained language models and their application for neural machine translation. For example, T5 (Raffel et al., 2020) uses Transformers and transfer learning to translate three languages at once. This provides the model with a rich vocabulary of about 32000 tokens, and it can use its prior knowledge to reach higher performances on languages for which there is less training data. However, as stated in the paper, the model can only process a predetermined, fixed set of languages and it uses a fixed vocabulary. This means that as much as it is able to infer information in general translation problems, it encounters the same OOV problem as other Seq2Seq-type models, since it does not have the ability to learn new words once training is over. Very recent concurrent efforts explore the use of pretrained language models for SPARQL query generation (Banerjee et al., 2022). For example, SGPT (Rony et al., 2022) is built on GPT-2 (Radford et al., 2019) and aims to generate SPARQL queries by encoding linguistic features of questions and the knowledge graph. It uses an entity masking strategy and generates queries with placeholders. After a query is generated by the neural architecture, a *post-processor* places the correct KB elements in the right places in the query. While our objective is similar, our approach aims at using a copy mechanism directly in the Seq2Seq architecture to place KB elements in the question instead of doing it in a post-processing step.

KGQA Systems. Since the handling of OOV KB elements is limited in the specific field of SPARQL NMT, it is necessary to broaden our research and learn from similar SPARQL NLP tasks. In particular, Knowledge Graph Question Answering (KGQA) systems aim to reconstruct a subgraph of the RDF schema from a natural language question and use it to generate a correct query. A notable aspect of these architectures is that they can provide a correct answer to a question on a topic not seen in training (Jiang and Usbeck, 2022) (if the answer is in the KB). An interesting KGQA system is HGNet (Chen et al., 2021b). A key aspect of this architecture is that in trying to generate the subgraph necessary to answer the question, it can take advantage of the fact that such graph often contain duplicated vertices. It uses LSTMs and a copy mechanism to copy these duplicated vertices, thus facilitating the generation task. Such systems (Chen et al., 2021b; Vollmers et al., 2021) highlight the importance of integrating the RDF schema and resources in the architecture. Doing so not only provides us with additional information on the KB elements themselves, but also on the elements which they are related to and which are more likely to be referenced as well.

SQL Systems. It is also useful to explore what we can learn from problems similar to the one of SPARQL NMT, such as the text-to-SQL seman-

[1] https://github.com/Lama-West/SPARQL_Query_Generation_aacl-ijcnl2022

tic parsing problem (Wang et al., 2020; Scholak et al., 2021). One of the current best performing model (Guo and Gao, 2019) is not a Seq2Seq-type model, but rather a classification model that learns to predict 6 different SQL components by leveraging the extensively annotated WikiSQL dataset. Seq2SQL (Zhong et al., 2017) is another approach, which, while not the best performing architecture, is worth noting for its schema integration mechanism. Seq2SQL augments the natural language question by concatenating it to all the columns' names and to the SQL vocabulary. The schema is essentially integrated directly in the input. Once again, incorporating the schema in the architecture gives the model enough information to understand which database elements (or for SPARQL, KB elements) are referenced in a question whether or not it has seen them during training, provided they are available in the database.

Copy Mechanism. The copy mechanism has shown its effectiveness in several encoder-decoder NLP tasks such as summarization (See et al., 2017), grammatical errors correction (Zhao et al., 2019), and knowledge graph question answering (KGQA) (Chen et al., 2021b). However, to our knowledge, it has not yet been used in SPARQL NMT as we propose here. Our hypothesis is that, given a working tagging algorithm where, in the NL question, mentions related to a KB element are replaced by their KB URI, a model could learn to copy the KB URIs from the question to the query instead of generating them. Notably, we propose to integrate CopyNet (Gu et al., 2016), whose copy mechanism comes after the decoder. For each token of the output sentence, it uses attention to calculate the probability that the token should be generated from the target vocabulary, and the probability that the token should be copied directly from the source. The chances of copying are slightly higher for OOV words in the source sentence.

Limitations. As reported by Yin et al. (2021), the current best performing non-pretrained architectures for SPARQL NMT are the *Transformer Seq2Seq* and the *ConvSeq2Seq*, which are Seq2Seq-type models where the encoder and decoder are respectively transformers and convolutional networks. As such, they encounter the same limitation as all Seq2Seq-type models, which is that because of the use of fixed vocabularies, the models are unable to fully handle OOV tokens. In SPARQL NMT systems, this results in the models not being able to answer questions referencing KB elements that were unseen during training. Instead, when encountering a question on a new KB element, the models generate a query referencing the element seen the most in the current context, even if it is not the one referenced in the question.

This also means that the model might learn the meaning of a specific KB element during training, but never use this knowledge if the element is not referenced in the test set. In the context of a query language, our hypothesis is that the encoder-decoder model should focus on learning the syntax of the correct SPARQL query related to a question, instead of trying to learn the meaning of each KB element. Keeping in mind that the prevalent KBs such as DBpedia can contain tens of thousands of different URIs, expecting the model to learn everything from examples is not optimal. Furthermore, the lack of real-world data is the field of SPARQL NMT makes this approach unrealistic.

In light of these limitations, the impressive BLEU-scores reported by Yin et al. (2021) raise some questions on the ability of these metrics and current datasets to properly evaluate NMT SPARQL models. Knowing that the models are only able to generate tokens learned during training, it is almost impossible for them to return a correct answer on a question whose topic is unknown, except by accident or when the expected answer is empty. Some datasets contain a number of queries that return empty answers. As such, it is important to make sure that models are thoroughly tested, especially on questions mentioning KB elements never seen during training.

3 Architectures

3.1 Base Architectures

This section describes the two best non-pretrained architectures for SPARQL NMT as reported by Yin et al. (2021), as well as our contribution.

ConvS2S. The convolutional sequence to sequence model (ConvS2S) is a Seq2Seq-type model where the encoder and decoder are convolutional networks (Gehring et al., 2017). Both the encoder and the decoder generate token embeddings and position embeddings of the vectors they receive as input, respectively the encoding of the question and the encoding of the query. The decoder also receives the output of the encoder as input, and its in-

put vector is padded at the beginning. This creates an offset which allows the model to learn from previous words and not from the current words which it is supposed to predict. Then, the sum of the token and position embedding vectors passes multiple times through a recurrent layer. This layer comprises a 1-dimension convolution and a Gated Linear Unit (GLU) in the encoder, followed by multi-head attention in the decoder. Following the survey by Yin et al. (2021), we use the same architecture configuration as FairSeq's *fconv_wmt_en_de* NMT architecture (Ott et al., 2019), described in Table 1.

Model	Transformer	ConS2S
Batch Size	128	128
Layers	6	15
Hid. Dim.	1024	[(512, 3) * 9, (1024,3) * 4, (2048, 1) * 2]
Dropout	0.5	0.2
LR	0.0005	0.5[2]
Optimizer	Adam	SGD

Table 1: Configuration of our Architectures

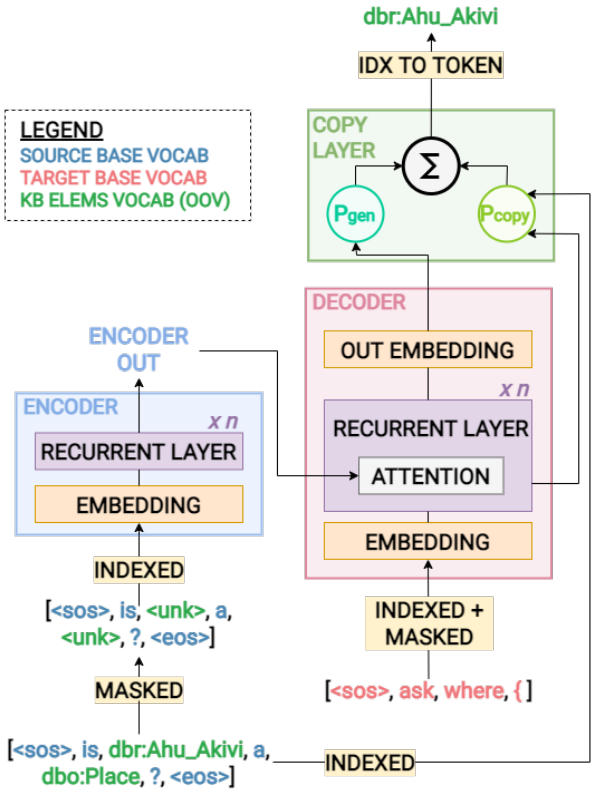

Figure 2: Encoder and copy-augmented decoder structure and interaction

Transformer. The Transformer model is a Seq2Seq-type model where the encoder and decoder are transformers (Vaswani et al., 2017). The encoder and decoder receive the same inputs as the ConvS2S. The decoder uses a multi-head attention layer that is not in the encoder. Our implementation is based on the FairSeq implementation (Ott et al., 2019) of the *transformer_iwslt_de_en* architecture, as described in Table 1.

3.2 A Copy-augmented Architecture

Figure 2 shows our generic architecture, which enriches any encoder-decoder model (e.g. CNNs or transformers) with a copy layer in the decoder. It generates specific source and target vocabularies that include the KB elements as explained below.

Vocabularies. In the baseline architectures (without copy), the source vocabulary comprises every token of the questions, and the target vocabulary comprises every token in the queries. Tokens are added in the order in which they are encountered.

However, when using the copy layer, there needs to be a way to differentiate tokens that are part of the base vocabularies (which the model will learn to generate) and tokens that are KB elements (which the model will learn to copy from the source). The latter are identifiable by their prefix, meaning tokens that start with **dbo:**, **dbr:**, **dbp:**, **dbc:**, **geo:**, **georss:** or **dct:**. Also, since the model receives vectors of indices and not words, tokens copied from the source to the target sentence must have the same index in both the source and target vocabularies.

To accommodate these constraints, we create a base source vocabulary and a base target vocabulary containing all tokens in the inputs but no KB elements and pad them with filler words so they are the same size. Then, we extract the KB elements in a vocabulary extension that contains all elements in both the questions and the queries. Finally, the KB vocabulary is concatenated to each base vocabulary to create our source and target vocabularies.

As we know the cutoff index of the initial vocabularies, we can quickly determine that each index above this cutoff represents a KB element we want to copy. During inference, if a new KB element is encountered, we can add it at the end of our source and target vocabularies, giving the model the capacity to copy it.

[2]For the dataset TNTSPA, we used a LR of 3.5

Copy Layer. In a copy-augmented architecture, the encoder and decoder receive masked source and target vectors, meaning any token above the cutoff index (and as such, out of the vocabulary) is replaced by a 0, representing an unknown token. As the role of the copy layer is to handle KB elements, this masking lets the encoder and decoder focus on the syntax rather than on the KB elements.

The copy layer comes after the decoder. It takes as input the unmasked encoded question and the decoder output, comprised of the attention scores and the probability of generating each word of the base target vocabulary. Ported to the Transformer architecture by (See et al., 2017; Zhao et al., 2019), we were able to adapt it to ConvS2S since both generate multi-head attention scores.

First, we identify whether there are any KB elements amongst the tokens of the encoded question by using the cutoff index. If it is the case, we extend the output probability tensor to include these extra tokens and initially assign them a generation probability of 0. Then, we calculate the probability of each token being generated, which is the softmax of the probability tensor. Using the attention score, we also calculate the probability of each word being copied directly from the source sentence. Following the implementation of (Zhao et al., 2019), we compute a balancing factor $\alpha_{bal} \epsilon [0, 1]$ between the copy and the generation probabilities using Equation 2, where Q, K and V are the query, key and value needed to calculate attention and W^{T} is a learnable parameter. The final probability of each token being the next word is the sum of the generation and copy probabilities balanced by this factor.

$$A_t = Q^T * K \tag{1}$$

$$\alpha_{bal} = sigmoid(W^T * (A_t^T * V)) \tag{2}$$

4 Methodology

4.1 Datasets

Format. Most natural language (NL) to SPARQL datasets are generated using templates to compensate for the lack of real-world data. A template is an NL question and its corresponding SPARQL query, in which there are annotated blanks to indicate the types of the KB element to insert (resources, classes, properties). These blanks are then replaced by KB elements' labels in the questions, and KB URIs in the queries. Many datasets also use an alternate version of SPARQL introduced by (Soru et al., 2017) called *intermediary SPARQL*, in which each symbol (e.g., brackets, dots) is replaced by a specific natural language expression. This encoding aims to make SPARQL closer to a natural human language. URIs are also reduced using their prefixes. To return to the original executable SPARQL query, one only has to make the inverse permutations. Table 2 shows the datasets used in this work. We split the datasets in an 80-10-10 fashion to reproduce the results reported by (Yin et al., 2021).

	Mon	**Mon50**	**Mon80**
Train	1797	1787	1791
Test	815	825	816
Int. rate	0.928	0.925	0.925
	TNTSPA	**LC-QuAD**	**DBNQA**
Train	4153	4150	145 429
Test	1045	1066	38 348
Int. rate	0.704	0.713	0.797

Table 2: Summary of the distribution of KB elements in the datasets

Monument. The Monument dataset (Soru et al., 2017) consists of pairs of English natural questions and intermediary SPARQL queries generated from 38 templates. The authors (Yin et al., 2021) generate other versions of the dataset: Monument, Monument50 and Monument80. The three versions are very similar in that they are all generated using 600 examples per template with different combinations of KB elements. We used their versions to be able to compare our results to state-of-the-art architectures. The high BLEU scores reported by Yin et al. (2021) are explained by the fact that most KB elements in the test set have already been seen during training, as shown by the high intersection rate in Table 2. Also, this dataset covers fewer KB elements in more entries, which gives the models plenty of examples to learn each element in its context. Overall, good results on this dataset only mean a model is functional.

LC-QuAD. The LC-QuAD datasets provide entries of multiple types (COUNT, ASK, SELECT) and cover a broad range of KB elements. We prioritized LC-QuAD v1.0 (Trivedi et al., 2017) over the newer LC-QuAD v2.0 (Dubey et al., 2019) since the models to which we compare our work are

trained on the first version. Further tests on LC-QuAD 2 are left for future work.

In LC-QuADv1.0, each entry contains an English natural language question and its corresponding SPARQL query generated from a template (called intermediary question), as well as a version of the question reformulated by an expert (called corrected question). It comprises 5000 entries generated from 33 of the 43 templates available. Table 2 shows that it is much more challenging than Monument. Indeed, there are many more different KB elements, fewer examples per element, and a lower intersection rate between the train and test sets.

We use three versions of the LC-QuAD dataset. The first version, referred to as **LC-QuAD Intermediary Questions**, uses the intermediary questions and their corresponding queries. These questions use the formulations defined by the templates. The second and more challenging version, referred to as **LC-QuAD Corrected Questions**, uses the reformulated natural language questions of the dataset and their corresponding queries. The third version, referred to as **TNTSPA**, is the version generated by the authors of the survey (Yin et al., 2021). It contains the reformulated questions (formulated in a more natural way) and queries found in the LC-QuADv1.0 dataset, but is split differently. Since no validation set is provided for the TNTSPA dataset, we use entries from LC-QuAD v1.0 that are not in the TNTSPA train or test sets. Since there are no templates associated to this dataset, we only use it to ensure we are able to reproduce state-of-the-art results with our implementation of the baselines architectures.

DBNQA. The DBpedia Neural Question Answering (DBNQA) dataset (Hartmann et al., 2018) is composed of 894,499 pairs of natural language questions and SPARQL queries. The entries are generated using 5165 question-query templates, constructed from entries in the LC-QuADv1.0 (Trivedi et al., 2017) and QALD-7 (Usbeck et al., 2017) datasets. We used the templates provided with the dataset but we did not manage to match all entries. We then extracted and corrected 512 templates suitable for the annotation of the questions and used the 398,284 entries corresponding to these templates. We also provide directly executable SPARQL queries instead of intermediate SPARQL queries.

RDF schema integration. As this research focuses mainly on finding a solution for the OOV problem, we developed a rudimentary tagging algorithm that leverages the templates. For each entry, we replace the KB elements labels that replace the blanks in the questions with their corresponding URIs in the query. KB elements that would be encoded as multiple tokens because of intermediary SPARQL (e.g., **[dbr_Cenotaph_, attr_open, Montreal, attr_close]**) are encoded as a single token (e.g., **dbr_Cenotaph_(Montreal)**) to reduce the vocabulary size. This dependence on templates is why we use the LC-QuAD Intermediary Questions version of LC-QuAD to train and evaluate our copy-augmented models, as it is the only version we could tag with complete accuracy. Figure 3 shows an entry before and after tagging.

Template: what is the **<domain>** whose **<property_1>** is **<resource_1>** and **<property_2>** is **<resource_2>** ?

Question: what is the **formula one racer** whose **relatives** is **ralf schumacher** and **has child** is **mick schumacher** ?

Tagged: what is the **dbo:FormulaOneRacer** whose **dbp:relatives** is **dbr:Ralf_Schumacher** and **dbo:child** is **dbr:Mick_Schumacher** ?

Figure 3: A tagged question

OOV Datasets. Finally, we generate an additional test set of 250 entries for each dataset called the **OOV Set**. First, we go through the dataset and make a list of all the referenced KB elements. Then, we use the templates to generate entries where the placeholders are replaced by KB elements that are not in the list, effectively creating a dataset in which no KB element has been seen in training.

To avoid false positives, we built our datasets so that questions would return a non-empty answer whenever possible. However, this proved to be a challenging task and our most successful attempts still contain about 70% of empty answers (count of 0, ask that returns false, or empty sets of elements). False positives can happen when a query returns an empty answer regardless of the KB elements referenced (e.g. an impossible question that links unrelated KB elements, or a question for which the KB does not contain an answer).

4.2 Evaluation

We use two main metrics to evaluate the original test sets and the oov test sets: the **BLEU-score** and the **answer accuracy**, which calculates the accuracy of the answers returned by the generated queries against the expected answers.

5 Results

We trained and evaluated our implementation of the models using Google Colab GPUs. We compare our results to those reported by (Yin et al., 2021), who train their model on HPC servers using the FairSeq implementations of the CnnS2S and Transformer architectures. It is important to note that they report the peak performance while we report the average of three runs. This means that we expect slightly lower performances when reproducing their results.

Baseline architectures on original datasets. Table 3 shows the results of the baseline architectures on original datasets. We clearly reproduce the performances of the survey by Yin et al. (2021). Even if our results for LC-QuAD are slightly lower, it is still within an acceptable margin. Because of the randomness of the weights initialization, the performance difference between a good and a underperforming run can be up to ten points. This margin also accounts for the small difference between TNTSPA and the LQ Corr Qsts. The higher scores on LQ Intrm.Qsts compared to the corrected questions are explained by the fact that the questions are generated from templates. This results in a smaller source vocabulary compared to the vocabulary of reformulated questions (used in TNTSPA and Corr. Qsts), since the questions are all formulated using the same template-words. Hence, the reduced variance helps the model understand the questions better.

Baseline architectures on tagged datasets. Table 4 shows the results of the baseline architectures on tagged datasets. We must not overlook the fact that using tagged data might help the architectures perform better, even without a copy layer. Since the KB elements are encoded as a single symbol, the size of the source and target vocabularies decreases, which usually helps the models perform better. These changes do not make much difference for the Monument datasets since the datasets contain enough examples for the models to learn the KB elements with or without tagging. For the LC-QuAD intermediary questions, we see a clear increase in performance. This is explained by the fact that in the untagged version, the URIs are encoded in the SPARQL query using multiple tokens (`dbr:Primus_ attr_open band attr_close`), whereas they are encoded as a single token in the NL question and the SPARQL query in the tagged version (`dbr:Primus_(band)`).

	Transformer		ConvS2S	
Dataset	BLEU	Acc.	BLEU	Acc.
Mon	95.86	90.55	96.35	91.66
Mon50	96.26	91.72	95.25	88.34
Mon80	96.35	92.69	94.47	82.68
TNTSPA	55.98	42.80	52.24	44.00
Corr. Qsts	49.61	32.07	49.94	40.80
Intrm. Qsts	60.31	43.60	65.65	47.40
DBNQA	64.86	46.41	67.26	45.43

Table 3: Performances of baseline architectures on original datasets. **TNTSPA** is (Yin et al., 2021)'s version of LC-QuAD. **C. Qsts** designates the LC-QuAD corrected questions and **Intrm. Qsts** designates the LC-QuAD intermediary questions.

For DBNQA, many URIs are quite long and expressed using multiple tokens in the questions. In the untagged version, this means many NL tokens are reused across multiple URI expressions, resulting in a smaller source vocabulary of 99603 tokens. Because there are more unique URIs than unique NL tokens used to represent these URIs in the questions, the tagged version uses a bigger source vocabulary composed of 158014 tokens. However, we see by comparing tables 3 and 4 that this augmentation of the vocabulary size does not affect the performance of the baseline models.

Copy-augmented architectures. Table 4 shows the results of our copy-augmented architectures on tagged datasets. We observe a strong increase in performance for LC-QuAD and DBNQA, which is impressive considering the number of different KB elements in the datasets, as well as perfect results on the Monument datasets. However, the most telling results are those obtained on the OOV datasets, reported in Table 5. The answer accuracy metric is not included because of the high proportion of possible false positives across all OOV datasets. Still, using only the BLEU score, we see that the baseline architectures struggle to han-

	Mon		Mon50		Mon80		Intrm. Qsts		DBNQA	
Architecture	BLEU	Acc.	BLEU	Acc.	BLEU	Acc.	BLEU	Acc.	BLEU	Acc.
Transformer	97.02	92.81	97.41	94.41	97.80	94.86	70.29	51.93	65.63	47.75
Transf.-copy	100	100	100	100	100	100	98.38	97.60	93.88	85.09
ConvS2S	97.82	95.26	97.71	95.13	98.14	95.96	76.62	52.93	67.57	45.22
ConvS2S-copy	100	100	100	100	100	100	98.35	97.40	95.40	86.87

Table 4: Performances of all architectures on tagged datasets

dle KB elements they have never seen, which is more representative of the actual capabilities of the models. Similarly, the results on tagged OOV datasets with baseline architectures are still low compared to the results on the original test sets, since tagged data still does not allow the model to adequately handle new KB elements after training. However, on copy-augmented architectures, we observe perfect performances on Monument, representing an increase in performance of about 30 BLEU points compared to its baseline counterpart. On LC-QuAD, the increase of about 40 BLEU points shows that the models handle better unknown KB elements using a copy mechanism.

6 Discussion

In view of these results, it is clear that, given a working tagging mechanism, the use of a copy-augmented architecture is an excellent advantage for SPARQL NMT architectures as it allows them to handle KB elements not seen in training. Furthermore, comparing the results with and without copy reported in Table 5, we see a clear improvement in the quality of the translations.

Another advantage of using a copy-augmented architecture is that it can perform almost as well on small datasets as on larger ones, as demonstrated by the high performances on the LC-QuAD Intermediary Questions and DBNQA. Essentially, the model does not need to learn the correspondences between each expression and the related URI anymore, and it does not need as many examples to learn the templates' formulations since there are not that many. Our work also highlights, as shown by the drastic difference between tables 3 and 5, that baseline models that are reported to have almost perfect performance are, in fact, not as effective outside the test set on which they are evaluated. Even if the BLEU score is a good way to evaluate the quality of the translation, The use of accuracy and the introduction of OOV datasets helps us understand better a model's actual capabilities.

There is however still room for improvement. Some of the limitations of this research lie in the use of template-based entries. In its current state, our copy-augmented architecture depends on questions following specific templates. As shown by the results reported in table 3, Seq2Seq models seem quite efficient at learning templates. As we see in Table 4, the performances increase when the KB elements are encoded in the questions, hinting at the fact that the model is limited by the large amount of KB elements in the dataset rather than the questions' formulations. Moving away from template-based datasets would also allow us to determine whether the copy layer helps the model understand the underlying schema of the KB.

We also need to improve the way OOV datasets are generated to be able to get a representative accuracy metric that is not biased by false positives. To do so, we must ensure most - if not all - queries return a non-empty answer.

Finally, another limitation is that our copy-augmented models depend on tagged questions to reach their top performance.

7 Conclusion

This paper determined that, coupled with a copy-augmented architecture, integrating the KB elements directly in the questions is sufficient for a SPARL NMT model to handle OOV KB elements and to obtain a significant increase in performance. These tagged datasets were used to train baseline and copy-augmented versions of the Transformer and the ConvS2S architectures. Using a copy layer, we report perfect performances on the Monument dataset and the generated OOV Monument dataset. For LC-QuAD, we report an increase in BLEU score of 20 points and an increase in answer accuracy of about 40 points. For DBNQA, our results

	Monument		LQ Intrm. Qsts		DBNQA	
Dataset	Original	Tagged	Original	Tagged	Original	Tagged
Transf	60.16	65.55	51.50	56.75	40.92	41.19
Transf-copy	-	100	-	85.68	-	79.82
ConvS2S	63.88	48.31	55.85	60.98	40.62	40.66
ConvS2S-copy	-	100	-	90.16	-	89.13

Table 5: BLEU scores of all the models on the OOV datasets.

show an increase in BLEU score of 35 points on average, as well as an increase in answer accuracy of 40 points. Our future work will involve the design of a neural tagging model and a joint tagging objective for our Seq2Seq models, as well as the comparison of our copy-augmented models with large pre-trained models and the use of these models as our encoders-decoders. Notable models on which to test our methodology include T5 (Raffel et al., 2020), BART (Lewis et al., 2020) and GPT-3 (Brown et al., 2020), as well as models that can generate code such as Codex (Chen et al., 2021a).

Acknowledgements

This research has been funded by the NSERC Discovery Grant Program.

References

Debayan Banerjee, Pranav Ajit Nair, Jivat Neet Kaur, Ricardo Usbeck, and Chris Biemann. 2022. Modern baselines for SPARQL semantic parsing. In *SIGIR '22: The 45th International ACM SIGIR Conference on Research and Development in Information Retrieval, Madrid, Spain, July 11 - 15, 2022*, pages 2260–2265. ACM.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, pages 1877–1901. Curran Associates, Inc.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harrison Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Joshua Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. 2021a. Evaluating large language models trained on code. *CoRR*, abs/2107.03374.

Yongrui Chen, Huiying Li, Guilin Qi, Tianxing Wu, and Tenggou Wang. 2021b. Outlining and filling: Hierarchical query graph generation for answering complex questions over knowledge graph. *CoRR*, abs/2111.00732.

Mohnish Dubey, Debayan Banerjee, Abdelrahman Abdelkawi, and Jens Lehmann. 2019. Lc-quad 2.0: A large dataset for complex question answering over wikidata and dbpedia. In *The Semantic Web - ISWC 2019 - 18th International Semantic Web Conference, Auckland, New Zealand, October 26-30, 2019, Proceedings, Part II*, volume 11779 of *Lecture Notes in Computer Science*, pages 69–78. Springer.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. 2017. Convolutional sequence to sequence learning. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 1243–1252. PMLR.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O. K. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*, pages 1631–1640. The Association for Computer Linguistics.

Tong Guo and Huilin Gao. 2019. Content enhanced bert-based text-to-sql generation. *CoRR*, abs/1910.07179.

Ann-Kathrin Hartmann, Tommaso Soru, and Edgard Marx. 2018. Generating a large dataset for neural question answering over the dbpedia knowledge base.

Longquan Jiang and Ricardo Usbeck. 2022. Knowledge graph question answering datasets and their generalizability: Are they enough for future research? In *SIGIR '22: The 45th International ACM SIGIR Conference on Research and Development in Information Retrieval, Madrid, Spain, July 11 - 15, 2022*, pages 3209–3218. ACM.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 7871–7880. Association for Computational Linguistics.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Demonstrations*, pages 48–53. Association for Computational Linguistics.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21:140:1–140:67.

Md. Rashad Al Hasan Rony, Uttam Kumar, Roman Teucher, Liubov Kovriguina, and Jens Lehmann. 2022. SGPT: A generative approach for SPARQL query generation from natural language questions. *IEEE Access*, 10:70712–70723.

Torsten Scholak, Raymond Li, Dzmitry Bahdanau, Harm de Vries, and Chris Pal. 2021. Duorat: Towards simpler text-to-sql models. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021*, pages 1313–1321. Association for Computational Linguistics.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 1073–1083. Association for Computational Linguistics.

Tommaso Soru, Edgard Marx, Diego Moussallem, Gustavo Publio, Andre Valdestilhas, Diego Esteves, and Ciro Baron Neto. 2017. SPARQL as a foreign language. In *Proceedings of the Posters and Demos Track of the 13th International Conference on Semantic Systems - SEMANTiCS2017 co-located with the 13th International Conference on Semantic Systems (SEMANTiCS 2017), Amsterdam, The Netherlands, September 11-14, 2017*, volume 2044 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Priyansh Trivedi, Gaurav Maheshwari, Mohnish Dubey, and Jens Lehmann. 2017. Lc-quad: A corpus for complex question answering over knowledge graphs. In *The Semantic Web - ISWC 2017 - 16th International Semantic Web Conference, Vienna, Austria, October 21-25, 2017, Proceedings, Part II*, volume 10588 of *Lecture Notes in Computer Science*, pages 210–218. Springer.

Ricardo Usbeck, Axel-Cyrille Ngonga Ngomo, Bastian Haarmann, Anastasia Krithara, Michael Röder, and Giulio Napolitano. 2017. 7th open challenge on question answering over linked data (QALD-7). In *Semantic Web Challenges - 4th SemWebEval Challenge at ESWC 2017, Portoroz, Slovenia, May 28 - June 1, 2017, Revised Selected Papers*, volume 769 of *Communications in Computer and Information Science*, pages 59–69. Springer.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008. Curran Associates, Inc.

Daniel Vollmers, Rricha Jalota, Diego Moussallem, Hardik Topiwala, Axel-Cyrille Ngonga Ngomo, and Ricardo Usbeck. 2021. Knowledge graph question answering using graph-pattern isomorphism. In *Further with Knowledge Graphs - Proceedings of the 17th International Conference on Semantic Systems, SEMANTiCS 2017, Amsterdam, The Netherlands, September 6-9, 2021*, volume 53 of *Studies on the Semantic Web*, pages 103–117. IOS Press.

Bailin Wang, Richard Shin, Xiaodong Liu, Oleksandr Polozov, and Matthew Richardson. 2020. RAT-SQL: relation-aware schema encoding and linking for text-to-sql parsers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 7567–7578. Association for Computational Linguistics.

Xiaoyu Yin, Dagmar Gromann, and Sebastian Rudolph. 2021. Neural machine translating from natural language to SPARQL. *Future Gener. Comput. Syst.*, 117:510–519.

Wei Zhao, Liang Wang, Kewei Shen, Ruoyu Jia, and Jingming Liu. 2019. Improving grammatical error correction via pre-training a copy-augmented architecture with unlabeled data. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 156–165. Association for Computational Linguistics.

Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2sql: Generating structured queries from natural language using reinforcement learning. *CoRR*, abs/1709.00103.